Design of a Multi-Context FPVLSI based on an Asynchronous Bit-Serial Architecture

Waidyasooriya Hasitha Muthumala, Masanori Hariyama and Michitaka Kameyama
Graduate School of Information Sciences, Tohoku University
Aoba 6-6-05, Aramaki, Aoba, Sendai, Miyagi,980-8579, Japan
Email: {hasitha@kameyama., hariyama@, kameyama@}ecei.tohoku.ac.jp

***Abstract*— This paper presents a novel asynchronous bit-serial architecture for multi-context field programmable VLSIs (MC-FPVLSI). Conventional MC-FPVLSIs use global wires to distribute the context-ID signal. As a result, hardware utilization ratio decreases, since it is impossible to execute different contexts simultaneously. They also have a high power consumption and high area overhead due to the clock tree and context ID trees. The proposed MC-FPVLSI eliminates the clock tree and global context ID trees completely. It uses a locally distributed context-ID signal and therefore, partial reconfiguration and simultaneous execution of different contexts are possible. It also uses the same wires to transfer the data and context ID signal, so that the area can be reduced further. The proposed architecture is designed using 6-metal 1-poly 90nm CMOS process technology.**

***Keywords*— dynamically reconfigurable, FPGA, self timing**

I. Introduction

The dynamically reconfigurable field programmable VLSIs (DR-FPVLSI) provide more cost-effective implementations than the conventional field programmable gate arrays (FPGA). One of the typical DR-FPVLSI architecture is a multi-context FPVLSI (MC-FPVLSI). It has multiple memory bits belong to different configuration planes. The context-ID signal selects one of them as the configuration bit. A structure of an MC-FPVLSI is shown in Fig.1. MC-FPVLSIs are said to have a high hardware utilization ratio, because their interconnections are shared between contexts and they can be reconfigured as different circuits in real time. However, since they have only global context-ID wires, the context-ID of the whole chip has to be changed at the same time. As a result, each context has to wait until the execution of its previous context is completed, even though there are adequate and unused hardware resources. Therefore, it is very difficult to achieve a high hardware utilization ratio. The context-ID wires are dedicated wires that connected to every logic block and switch block in the whole chip. Therefore, they need a structure close to a typical clock tree to achieve fast context switching. When the number of context-IDs are increasing, more context-ID wires are needed and it leads to a high area overhead and high power consumption. To solve these problems, we propose a novel MC-FPVLSI based on an asynchronous bit-serial architecture, which can be reconfigured in partially using locally distributed context-ID signals. To the best of the authors knowledge, this is the first ever implementation of an asynchronous MC-FPVLSI architecture.

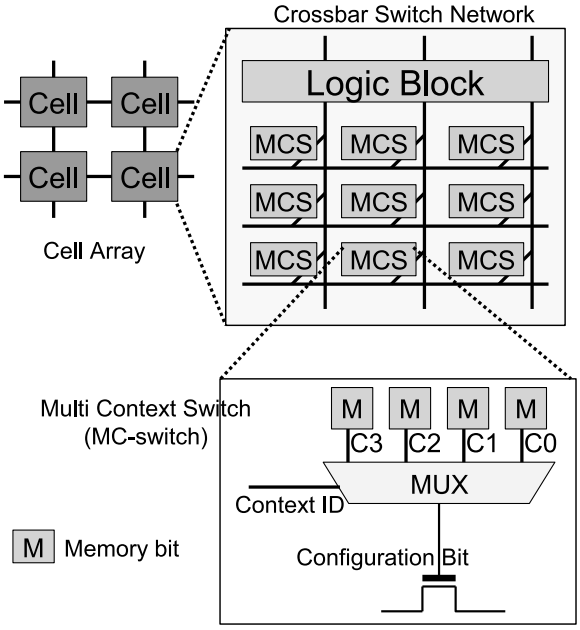

Fig. 1. Structure of a conventional MC-FPVLSI

In this paper we target the following areas in the conventional MC-FPGAs.

- Low hardware utilization ratio due to the global context-IDs that prevents partial reconfiguration.
- High static and dynamic power consumption caused by the clock trees and context-ID trees.
- High area overhead of the context-ID trees when the number of contexts are large.

We propose an asynchronous architecture to eliminate the clock tree completely. Then we remove the global context-ID trees by adopting a bit-serial architecture which can distribute the context-ID signals locally. As a result, context-IDs can be changed partially, so that different contexts can be executed simultaneously in different areas on the chip to increase the hardware utilization ratio. To reduce the area further, we use the same data lines to transfer the context-ID signals. Moreover, the proposed MC-FPVLSI can transfer any number of context-ID-bits without any additional interconnection overhead, unlike the conventional one that requires more context-ID lines when the number of contexts are increased.

978-1-4244-1680-6/07/$25.00 ©2007 IEEE

II. Architecture

A. Data and context-ID encoding scheme

Asynchronous encoding schemes can be classified into;

- Bundled-data encoding
- Delay-insensitive encoding

Fig.2 shows the overall architecture of the bundled-data encoding. It splits the data and request into separate wires. Data are sent similar to the synchronous architecture, where n-bits of data needs n number of wires. However, an explicit delay is inserted to the request signal to ensure that the request is received only after the valid data are available. This method needs only $(n+2)$ wires to send n-bits of data, therefore it can be implemented by using a relatively small area overhead. However, it requires the delay values to operate reliably. If the data path is fixed, it is possible to determine the delay constraint. However, in FPVLSIs, it is not easy to always meet the delay constraint since the data path is programmable.

Therefore we choose delay-insensitive encoding [2], since it does not require any delay insertion. In the delay-insensitive encoding, two wires are used for each data bit as data and request signals, unlike the bundled-data encoding where the request signal is shared between all the data bits. We use level encoded 2-phase dual-rail encoding (LEDR) to encode the data and request signals, since it is very fast data transfer method among the delay-insensitive encoding schemes [3]. Fig.3 shows the LEDR encoded signals (V, R) and their decoded values, data and phase. Fig.4 shows the data transfer based on LEDR encoding. The data bit equals to V and the phase equals to the XOR of V and R $(V \oplus R)$.

Since we want to use the same wires to transfer both data and context-ID signals, we need an another signal to separate the context-ID bits from the data bits. For this purpose, we use a new signal denoted by C as shown in Fig.5. In this paper, we use the term "CVR encoding" for this encoding scheme, since it uses 3 signals C, V and R to transfer a data bit or a context-ID bit. The encoded signals and their decoded values are shown in Fig.5. The phase equals to $[C \oplus V \oplus R]$ and the data bit equals to V. If the phase is changed, it indicates a new signal is received. If C is "0", the current bit is a data bit, and otherwise it is a context-ID bit.

An example of a bit-serial data and context-ID transfer is shown in Fig.6. At the beginning 3 data bits [0 0 1] are received in a bit-serial manner. Then the value of C changes from "0" to "1" indicating that the next bit is a context-ID-bit. After that, the context-ID-bits are received one after another. In this example, the received context-ID equals to 3 [0,1,1]. After all 3 context-ID-bits are received, the value of C changes from "1" to "0" indicating the next bit is a data bit.

B. Architecture of the logic block

Asynchronous cell based on bundled-data encoding is proposed in [4] for single context FPGAs. However, our proposed architecture is based on delay insensitive encoding and can handle multiple contexts. The block diagram of a logic block

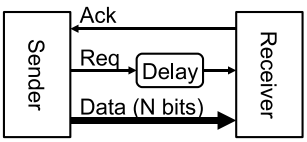

Fig. 2. Bundled-data encoding

Encoded signal		Data	Phase
V	R		
0	0	0	0
1	1	1	0
0	1	0	1
1	0	1	1

Fig. 3. LEDR encoding

Fig. 4. Data transfer based on LEDR encoding

Encoded signal			Data	Context ID	Phase
C	V	R			
0	0	0	0	-	0
0	1	1	1	-	0
0	0	1	0	-	1
0	1	0	1	-	1
1	0	0	-	0	1
1	1	1	-	1	1
1	0	1	-	0	0
1	1	0	-	1	0

Fig. 5. CVR encoding

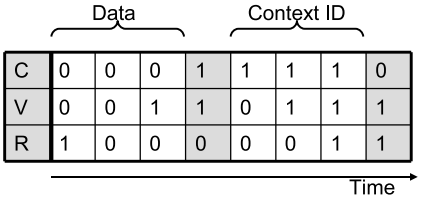

Fig. 6. Data transfer based on CVR encoding

(LB) in the proposed MC-FPVLSI is shown in the Fig.7. The LB has 3 main components;

1) Receiver
2) Transmitter
3) Multi-context look-up-table (LUT)

The receiver circuit receives the C, V, R values from previous LBs. When both input C values are "0", receiver passes the V, R values to the LUT. After the C values change from "0" to "1", the context-ID values are stored in the shift register in the receiver. After a context-ID bit is saved in the shift register, receiver sends the acknowledge (ACK) signals to the previous LBs, indicating it is ready to receive another bit. After all the context-ID bits are received, the transmitter starts sending the appropriate V, R values of the context-ID bits to the multiplexers. The multiplexers select the LUT output when C is "0" and the transmitter output when C is "1". The control

signal of the multiplexer is delayed for one cycle, so that it can select data and context-ID accurately as shown in Fig.6.

The architecture of a 2-input-8-context LUT is shown in Fig.8. It has 4 memory bits per each context to function as a programmable 2-input logic gate. When the appropriate configuration plane is selected by the context-ID-bits and the multiplexers, the selected memory bits are passed to the decoders. The structure of the decoder-11 in Fig.8 is shown in Fig.9. When the phase of the both inputs ([Va,Ra] and [Vb,Rb]) are "0", both V and R are equal to the memory bit M11, so that the output phase equals to the input phase after the calculation. Similarly, when the phase is "1", V equals to M11 and R equals to $\overline{M11}$. Note that, when the phases are different, both V and R become high-impedance, so that the values in the latches are preserved. Also note that, in every case, both inputs have the same C value. If C is different, the inputs to the LUT will be blocked by the receiver, so that the previous state of the latch will remain unchanged.

C. Interconnection block based on context-ID decoding

We adopt the interconnection block architecture proposed in the paper [6] since it efficiently reduces the area by 50%. The main idea behind this architecture is the redundancy in the configuration data. Since more than 90% of the configuration data are redundant [5], they can be implemented by a simple switch structure smaller than the conventional multi-context switch. Fig.10 (a) shows the implementation of a switch when the configuration data are independent of the context-ID-bit. Fig.10 (b) shows the implementation when the configuration data depend on a single context-ID-bit (C1). This simplified switch is called a switch element (SE) and it is the basic unit in the interconnection block. A switch block (SB) consists with two SEs aligned vertically and horizontally, so that it can act as a pass gate or a crossbar switch.

Fig.11 shows an example of routing two logic blocks, when the configuration data depend on more than one context-ID-bit. In here, LB1 and LB2 are connected when the context-ID bits are [1,1] and are disconnected in all the other cases. Therefore, we program all the SBs in the desired path except the one near the LB1 as pass gates that are always on. Then we program the SB near the LB1 as a pass gate controlled by an input "S", which equals to $C0 \cdot C1$, so that we can realize the desired connections for all the 4 contexts. Several SBs are combined to create a context-ID-decoder, which generates the control signal "S" using $C0$ and $C1$ as inputs. A detailed description on routing is given in the paper [6].

Let us explain how the overall architecture works (Fig.12). At the beginning, context-ID-bits are passed to the LBs through the I/O ports. For example, in an 8-context FPVLI, an LB does not transfer the context-ID-bits until all the 3 bits are received. When all the context-ID-bits are available, the context-ID-decoder can create the paths to the adjoining LBs. When a path is created, the LB sends the context-ID-bits through the newly created path to the other LBs that are connected. After all the paths in the circuit are created, the data can be send through those paths. In the proposed

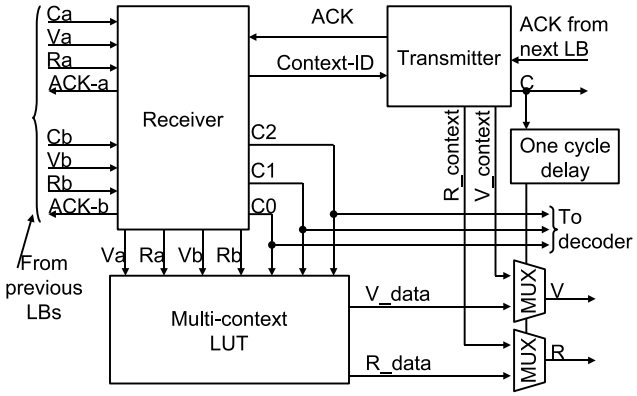

Fig. 7. Block diagram of a logic block

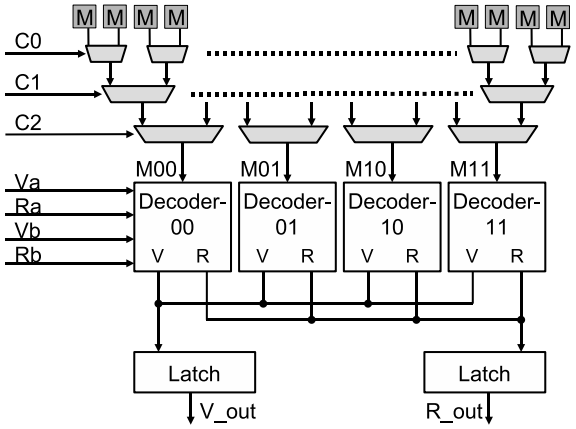

Fig. 8. Architecture of the multi-context LUT

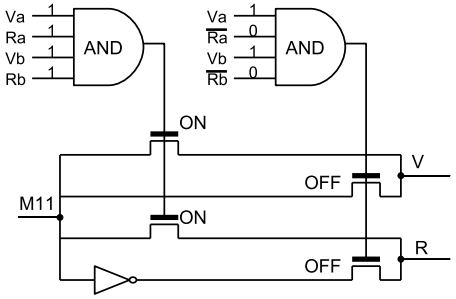

Fig. 9. Structure of the decoder-11

architecture, the context-ID-bits and the data bits are pipelined to minimize the context switching delay. To run a new context, the appropriate context-ID-bits are passed to the LBs, so that the desired circuit is created automatically. The scheduling of contexts, allocation and mapping should be done in advanced in the high-level-synthesis process.

III. Evaluation

The proposed MC-FPVLSI architecture is designed using 6-metal 1-poly 90nm CMOS design process. The layout of the chip is shown in Fig.13. The designed chip has 100 cells in a 10×10 cell array. Since the fabrication process of the actual chip is not completed yet, we have done the HSPICE circuit simulation for two near-by cells. According to the simulation, the minimum data transfer and the context ID transfer delays (for a single bit) are $1.32ns$ and $0.81ns$ respectively. Table I shows the performance of the chip.

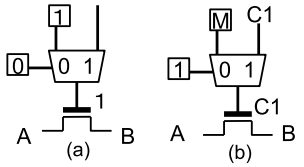

Fig. 10. Architecture of a switch element

Context ID		Connection between LB1 & LB2
C0	C1	
0	0	Disconnected
0	1	Disconnected
1	0	Disconnected
1	1	Connected

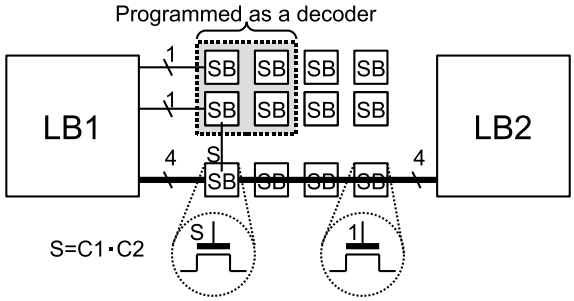

Fig. 11. Routing in proposed MC-FPVLSI

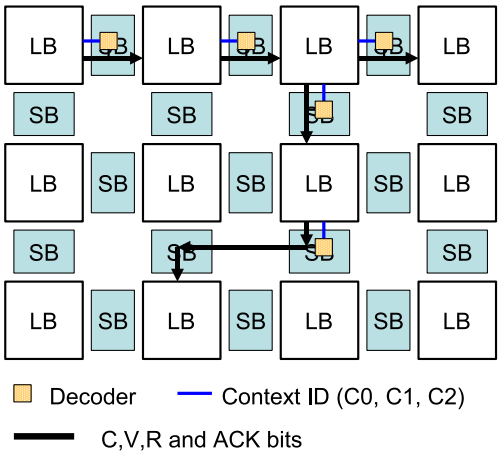

Fig. 12. Overall architecture

IV. Conclusion

We have proposed a novel MC-FPVLSI based on an asynchronous bit-serial architecture. Since local context-ID signals are used, the additional hardware needed to run large number of contexts is very small, unlike the conventional one, where it needs additional context-ID trees when the number of contexts are increased. Therefore, the proposed architecture is better suited for the circuits with large number of contexts.

Not only the area overhead, but also the hardware utilization is better in our architecture, since it can run multiple contexts simultaneously. Note that, when the number of contexts is high, the execution time and hardware utilization for each context varies, so that, even the most powerful high-level-synthesis tools may not be able to achieve a high utilization ratio in conventional MC-FPVLSIs.

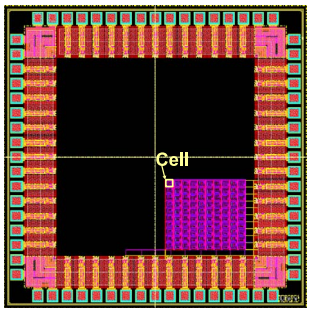

Fig. 13. The layout of the test chip

TABLE I

Performance of the proposed MC-FPVLSI

Number of contexts	8
Number of cells	100 (10 × 10)
Cell area	53 × 62 (μm^2)
Minimum context ID transfer delay (for single context-ID bit)	0.81 ns
Minimum data transfer delay	1.32 ns

The area of the proposed logic block is larger than that of the conventional one. However, since we are using an interconnection structure based on the context-ID decoding [6], the area of the interconnections is reduced to 50%. Note that, the area of the interconnetions accounts for more than 70% of the overall area in a MC-FPVLSI, so that the total area reduction is quite high.

Since we are not using any global clock or global context-ID signal, the peak current is considerably low, compared to the conventional one, where all the cells operate with the rise of a clock or a context-ID signal. As a result, low cost packages can be used in our design.

Acknowledgment

This work is supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with STARC, Fujitsu Limited, Matsushita Electric Industrial Company Limited., NEC Electronics Corporation, Renesas Technology Corporation, Toshiba Corporation, Cadence Design Systems Inc and Synopsys Inc.

References

[1] A. Dehon, "Dynamically Programmable Gate Arrays: A Step Toward Increased Computational Density", Proc. the Fourth Canadian Workshop on Field-Programmable Devices, pp.47-54, 1996.

[2] W. J. Bainbridge and S. B. Furber, "Delay insensitive system-on-chip interconnect using 1-of-4 data encoding", Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems, pp.118-126, March 2001.

[3] Mark E. Dean, Ted E. Williams and David L. Dill, "Efficient self-timing with level-encoded 2-phase dual-rail (LEDR)" Proc. 1991 University of California/Santa Cruz conference on Advanced research in VLSI, pp.55-70, 1991.

[4] C. Traver, R.B Reese and M.A. Thornton, "Cell designs for self-timed FPGAs" Proc. 14th Annual IEEE International ASIC/SOC Conference, pp.175-179, 2001.

[5] I. Kennedy, "Exploiting redundancy to speedup reconfiguration of an FPGA", Proc. 13th International Conference on Field Programmable Logic and Application, pp. 262-271, Sep. 2003.

[6] Weisheng Chong, M. Hariyama, and M. Kameyama, "Novel switch-block architecture using reconfigurable context memory for multi-context FPGAs", Proc. International Workshop on Applied Reconfigurable Computing (ARC 2005), pp.99-102, 2005.